# SYSTEM AND METHOD FOR MEASURING MIDDLEWARE RESPONSE TIME

## Cross-Reference to Related Applications

This application claims the benefit of U.S. Provisional Application No. 60/437,848, filed January 3, 2003, the entire disclosure of which is incorporated herein by reference.

## Field

The present application relates to message oriented middleware performance monitoring and, more particularly, to a system and a method for measuring middleware response time.

## Background Information

Generally, middleware is software that has its own application programming interfaces ("APIs") that help insulate software developers from operating system-specific APIs. A middleware layer is often located between client and server processes. By measuring the route time associated with the middleware layer of an application, problems can be identified and corrected resulting in improved performance and availability. Currently, the route time associated with a middleware



Express mail label no.: EJ622909925US Deposited: 30 DEC 2003

application can be measured by changing the code of the application itself. Such a procedure is considered intrusive. Other known procedures used for measuring route time rely on a host of resources and require synchronization among multiple components.

Thus, there is a need for a non-intrusive system and method that monitors the performance of middleware by measuring the route time and storage residency time, for example, the residency time of a message stored in at least one queue. As a result, a determination can made as to which component(s) of a configuration is problematic or potentially problematic. A need also exist for monitoring the middleware performance without requiring synchronization.

**Summary of the Invention**

An aspect of the present application provides for a method for monitoring middleware perfo rmance. The method may include determining a route time for a message transmitted along a predetermined network route, determining at least one queue residency time, the at least one queue residency time reflecting an amount of time at least one other mes sage is stored in at least one respective queue located along the predetermined

network route, and calculating a middleware response time according to the route time and the at least one queue residency time.

Another aspect of the present application provides for a method for monitoring middleware performance. The method may include determining a route time for a user-defined sample message to be transmitted along a predetermined network route, the route time reflecting an amount of time for the sample message to travel from an origination queue manager to a destination queue manager and then back along the same route to the origination queue manager, determining at least one queue residency time for at least one local production queue, the at least one local production queue being associated with the origination queue manager and/or the destination queue manager and the at least one queue residency time reflecting the amount of time an actual application message is stored in the at least one local production queue, and calculating a middleware response time by adding the route time to the at least one queue residency time.

A further aspect of the present application provides for a system for monitoring middleware performance. The system may include a computer system

adapted to generate a sample message and an application message, and a computer memory electrically connected to the computer system encoded with instructions for performing the following: determining a route time for the sample message transmitted along a predetermined network route; determining at least one queue residency time, the at least one queue residency time reflecting an amount of time the application message is stored in at least one respective queue located along the predetermined network route; and calculating a middleware response time according to the route time and the at least one queue residency time.

**Brief Description of the Drawings**

Fig. 1 illustrates an exemplary block diagram of the present disclosure for monitoring middleware performance;

Fig. 2 illustrates an exemplary flow diagram of an embodiment for determining route time, queue residency time and response time;

Fig. 3 illustrates an exemplary block diagram of the present disclosure for determining route time;

Fig. 4 illustrates an exemplary flow diagram of an embodiment for determining route time;

Fig. 5 illustrates an exemplary data structure;

Fig. 6 illustrates an exemplary block diagram of the present disclosure for determining queue residency time;

Fig. 7 illustrates an exemplary flow diagram of an embodiment for determining queue residency time;

Fig. 8a illustrates an exemplary user interface of the present disclosure for developing a route;

Fig. 8b illustrates an exemplary user interface of the present disclosure for monitoring route time;

Fig. 9a illustrates an exemplary user interface of the present disclosure for adding a queue threshold; and

Fig. 9b illustrates an exemplary user interface of the present disclosure for monitoring at least one queue.

**Detailed Description**

In the exemplary embodiments of the present application, application middleware response time for a message oriented middleware ("MOM") based application can be estimated without changes to the respective application(s) and is, therefore, non-intrusive, as described herein. An example of such a MOM based application is the e-business infrastructure software

sold by IBM® under the label WebSphere® MQ. WebSphere® MQ is a middleware solution for interconnecting business applications in heterogeneous environments. Middleware response time is described in the present application as being comprised of two components: route time and the sum of each of the storage residency times. Even though storage residency time is referred to in the present application as queue residency time, the exemplary embodiments are equally applicable to additional storage mediums besides queues. Additionally, application processing time is not considered in determining the middleware response time as described in detail herein. Alternatively, however, processing time can be included in the response time determination.

Figure 1 illustrates an exemplary MOM network 100. MOM network 100 includes three exemplary queue managers ("Qmgr") or MOM controllers, referred to herein as queue managers: queue manager 105a, queue manager 105b and queue manager 105c. Queue managers 105a, 105b, 105c are associated with node A 150a, node B 150b, node C 150c and local production queues 110a, 110b, 110c, respectively. Furthermore, queue managers 105a, 105b, 105c are associated with agents 115, 125, 130 and transmission queues 120a, 120b, 120c, respectively. The

number of queue managers in MOM network 100 and the number and type of queues associated with the respective queue managers are merely illustrative. Data can be transmitted from node A 150a, node B 150b and node C 150c to collection point 130. Collection point 130 includes memory unit 140 and processing unit 145.

Each transmission queue is associated with a corresponding remote queue definition, as can be seen in Fig. 1. Remote queue definitions are not real queues, but name aliases that are controlled as though they were real queues. An effect of the remote queue definition is to define a physical destination queue name and queue manager name. The use of remote queue definitions is well known to a person having ordinary skill in the art and is therefore not described in detail herein.

As will be described below, a sample message is transmitted from and to queue manager 105a along the same path or network route in MOM network 100 as would be used by an application to be monitored and the sample message is temporarily stored in each transmission queue 120a, 120b, 120c along the network route. The sample message is used to determine the route time. Additionally, an actual application message is stored at

each local production queue 110a, 110b, 110c to determine the respective queue residency times.

Figure 2 illustrates an exemplary flow diagram for estimating an application middleware response time of a MOM application. In order to measure the estimated response time for the MOM application, the route time and at least one queue residency time need to be determined. The exemplary equation for determining the application middleware response time ("AMRT") is as follows:

$$AMRT = \text{Route Time} + \Sigma\ (\text{Queue Residency Times}).$$

In order to determine the route time component of the equation, a sample message is generated, in 205. In an exemplary embodiment, the sample message includes a plurality of messages batched together. The size of each of the plurality of messages can be the same or have varying sizes. Alternatively, the sample message is a single message. Preferably, the sample message replicates an actual message that would be transmitted within MOM network 100. The number of messages batched in a sample message and the size of each of the messages

is user-controlled, for instance, by interacting with at least one user interface.

The sample message is stored, in 210. Agent 115 associated with queue manager 105a puts the sample message in transmission queue 120a. Transmission queue 120a temporarily stores the sample message to be sent to another part of the application residing on queue manager 105b. Remote queue definition associated with transmission queue 120a points to or identifies queue manager 105b and, more particularly, transmission queue 120b and its associated remote queue definition. Before the message is transmitted, the originating time is determined by agent 115, in 215. The originating time is stored in field 505 of sample message 500, as shown in Fig. 5, and the sample message is transmitted within MOM network 100, in 220.

Figure 4 sets forth in more detail the transmission of the sample message and the determination of at least one arrive time and at least sent time along the route. In 405, the sample message is transmitted from originating queue manager 105a to destination queue manager 105c via intervening queue manager 105b. Specifically, the sample message is first transmitted to and temporally stored in transmission queue 120b since

remote queue definition associated with transmission queue 120a points to transmission queue 120b. At node B 150b, channel or message exits associated with agent 125 determine timestamps for when the sample message enters transmission queue 120b and exits transmission queue 120b, referred to as the arrive time and the sent time, respectively, in 410. The arrive time and the sent time are added to the sample message in field 510a and field 510b, respectively, as shown in Fig. 5.

Next, the application component on queue manager 105b associated with agent 125 retrieves the sample message temporarily stored in transmission queue 120b and according to the respective remote queue definition transmits the sample message to another application component on destination queue manager 105c. At destination queue manager 105c, the sample message is temporarily stored in transmission queue 120c. Similarly, at node C 150c, channel or message exits associated with agent 130 determine timestamps for when the sample message enters transmission queue 120c and exits transmission queue 120c, referred to as the arrive time and the sent time, respectively, in 410. The arrive time and the sent time are added to the sample

message in field 515a and field 515b, respectively, as shown in Fig. 5.

The application component on queue manager 105c associated with agent 130 retrieves the message temporarily stored in transmission queue 120c and according to the respective remote queue definition transmits the sample message back to originating queue manager 105a via intervening queue manager 105b, in 415. In an exemplary embodiment, the sample message is transmitted back to originating queue manager 1 05a along the same route that the sample message traveled to get to destination queue manager 105c, a purpose of which is described below. The exemplary route of the sample message is shown in Fig. 3. Alternative routes for sample messages are also described below. In 420, like in 410, an arrive time and a sent time for the sample message are determined for intervening queue manager 105b. The arrive time and sent time are stored in field 520a and field 520b of the sample message, as shown in Fig. 5. Once the sample message arrives back to originating queue manager 105a, agent 115 determines an end time, in 225, and the end time is stored in field 525 of the sample message. In an exemplary embodiment, agent 115 determines that the sample message transmi tted

from queue manager 105a is the same as the received message by comparing an identification stored in a field of each received message.

Data indicating the various timestamps shown in Fig. 5 are transmitted to collection point 135. Processing unit 145 at collection point 135 determines the route time, in 230. Route time is defined as the time it takes a sample message to be transmitted from the queue manager where the message originated, for instance, queue managers 105a, across any intermediate queue manager(s), to a destination queue manager, for instance, queue manager 105c, and have the message transmitted back to the originating queue manager from the destination queue manager 105c along the same route. In an exemplary embodiment, measurement of route time relies on an independent process that sends sample messages through MOM network 100 using the same paths as the application to be measured. As described above, the route time component is measured by creating a series of daisy-chained remote queue definitions across all queue managers involved in the measurement, for example, queue managers 105a, 105b, 105c.

Based on the originating timestamp stored in field 505 and the end timestamp stored in field 525,

processing unit 145 determines the route time, in 230. Specifically, the route time is determined by taking the difference of the originating timestamp and the end timestamp. Since the originating timestamp and the end timestamp are associated with the same queue manager 105a, synchronization is not an issue. The route time is stored in memory unit 140 at collection point 135.

Furthermore, based on the arrive timestamps and the sent timestamps for intervening queue manager 105b and destination queue manager 105c stored in fields 510a...520b, intra-queue manager times are determined by processing unit 145. In particular, the difference between the arrive timestamp and the corresponding sent timestamp for a respective queue manager represents the intra-queue manager time. The sum of the intra-queue manager times along MOM network 100 represents the total queue manager processing time. In an exemplary embodiment, averages of the intra-queue manager times along MOM network 100 can also be calculated. Data representing the intra-queue manager times is also stored in memory unit 140.

In addition to determining the route time component, queue residency time is also needed for calculating the middleware response time, in 235. Figs.

6 and 7 illustrate queue residency time and how it is determined in an exemplary embodiment of the present application. Queue residency time is defined as the time that an actual application message is stored in local production queue 110a, in local production queue 110b and in local production queue 110c, as shown in Fig. 6. The sum of the queue residency times is referred to as the total queue residency time. In an exemplary embodiment, measurement of queue residency times relies on an independent process in which respective agents sample actual application messages being put on respective local production queues, identifies the sampled message through a channel message exit and captures the point when that message is pulled from the local production queue. In an exemplary embodiment, messages are sampled at an adjustable rate. Based on the sample set of messages, queue residency time is calculated as described below with reference to Fig. 7.

Agents 115, 125, 130 determine queue residency time for local production queues 110a, 110b, 110c, respectively. The procedure for calculating queue residency time is described herein only with reference to agent 115 and local production queue 110a since the

same procedure is followed at node B 150b and node C 150c in MOM network 100. As an application puts actual messages on the local production queue 110a, collection components such as a message exit and an API exit take at least one sample of the messages, in 705. Agent 115 determines and stores the identification ("ID") associated with the sampled message, in 710, and determines and stores the put timestamp, in 715 and 720, respectively. The put timestamp indicates the time that the sampled message was placed on local production queue 110a. In an exemplary embodiment, the ID is located in a field of the sampled message. When messages are retrieved from local production queue 110a by an application component, a comparison is performed between the IDs of the respective retrieved messages and the stored ID of the sampled message, in 725. If a positive match is made, another timestamp, referred to as the get timestamp, is determined, in 730. The get timestamp indicates the time the sampled message was retrieved from local production queue 110a. The queue residency time for local production queue 110a is then determined by agent 115, in 735. In an exemplary embodiment, the queue residency time equals the difference between the respective get timestamp and the respective put

timestamp. The calculated queue residency time is transmitted to collection point 135, in 740. The queue residency times associated with each queue manager 105a, 105b, 105c are transmitted to collection point 135. Processing unit 145 adds the queue residency times to determine the total queue residency time and stores the individual queue residency times and the total queue residency time in memory unit 140. In a further exemplary embodiment, each agent calculates respective queue residency times by sampling multiple messages stored in the queue, determining corresponding put timestamps and get timestamps for the sampled messages, calculate multiple queue residency times and average the queue residency times over an interval.

In order to determine the total queue residency time, queue managers 105a, 105b, 105c in MOM network 100 do not have to be synchronized since the difference(s) between the put timestamp(s) for a sampled message(s) and the get timestamp(s) for the sampled message(s) are determined for a respective local production queue.

The estimated application middleware response time, consisting of the time for the sample message to travel through the middleware network, referred to as the route time, and the time sampled messages reside in local

production queues 110a, 110b, 110c, is determined, in 240. The response time is then determined by combining the route time with the sum of queue residency times. Additionally, a network time can be determined by subtracting from the route time the total intra-queue manager time.

The exemplary embodiments of the present application for determining route time are described with reference to chained transmission queues that are kept independent from local production queues. By using transmission queues, the performance of actual application messages being transmitted over MOM network 100 is simulated. The exemplary embodiments, however, are equally applicable to using local production queues and/or actual application messages to determine route time and, thus, response time.

By transmitting data to collection point 135 and by storing this data in memory unit 140, additional performance information can be obtained, for instance, regarding queue statistics. Such queue statistics include average queue residency times, high and low queue residency times, number of get timestamps and the number of put timestamps. Further, thresholds can be set to generate alerts so that, for example, when a

queue residency time is greater than a threshold, an alert is transmitted for the respective queue.

Additionally, one or more user interfaces can be developed so that the performance of MOM network 100 can be monitored. A user can monitor the response time, queues residency time(s) and route time, whether these times are within predetermined parameters or exceed a threshold, the number of messages in the sample message and the size of each message. For instance, a user interface can be developed and used to monitor the performance of one or more queues.

Figure 8b illustrates an exemplary user interface for monitoring at least one queue of the present application. Further, Fig. 7b illustrates an exemplary user interface for monitoring route time of the present application. The user interface can include list of all the queues, status of each queue and thresholds and current value for each metric. The desired parameters and thresholds can by input by the user via one or more additional user interfaces. Figure 7a illustrates an exemplary user interface used by an agent for developing a route that a sample message will travel and Fig. 8a illustrates an exemplary user interface for adding a queue threshold for a particular queue.

In an exemplary embodiment, a route builder application is used for defining the path for the sample message. With the route builder application, each route is named and the individual queue managers are added. In the exemplary embodiments of the present application, the sample message is transmitted from originating queue manager 105a to destination queue manager 105c via intervening queue manager 105b and back to originating queue manager 105a along the exact same path, as shown in Fig. 3. Accordingly, originating queue manager 105a, intervening queue manager 105b and destination queue manager 105c do not have to be synchronized to determine route time. Alternatively, by changing the remote queue definition associated with destination queue manager 105c, the sample message can travel a different path back to originating queue manager 105a. Similarly, since the sample message is returning to originating queue manager 105a, the queue managers do not need to be synchronized. However, if the sample message is returned to a queue manager other than originating queue manager 105a, that queue manager and originating queue manager 105a need to be synchronized to determine the route time.

In a further alternative embodiment, a uni-directional route time can be determined, that is the time for the sample message to travel from originating queue manager 105a to destination queue manager 105c. Originating queue manager 105a and destination queue manager 105c would need to be synchronized to accurately estimate the route time. Data is collected at collection point 135 from the respective queue managers and route time is determined by processing unit 145.

The embodiments described above are illustrative examples of the present application and it should not be construed that the present application is limited to these particular embodiments. Various changes and modifications may be effected by one skilled in the art without departing from the spirit or scope of the invention as defined in the appended claims.